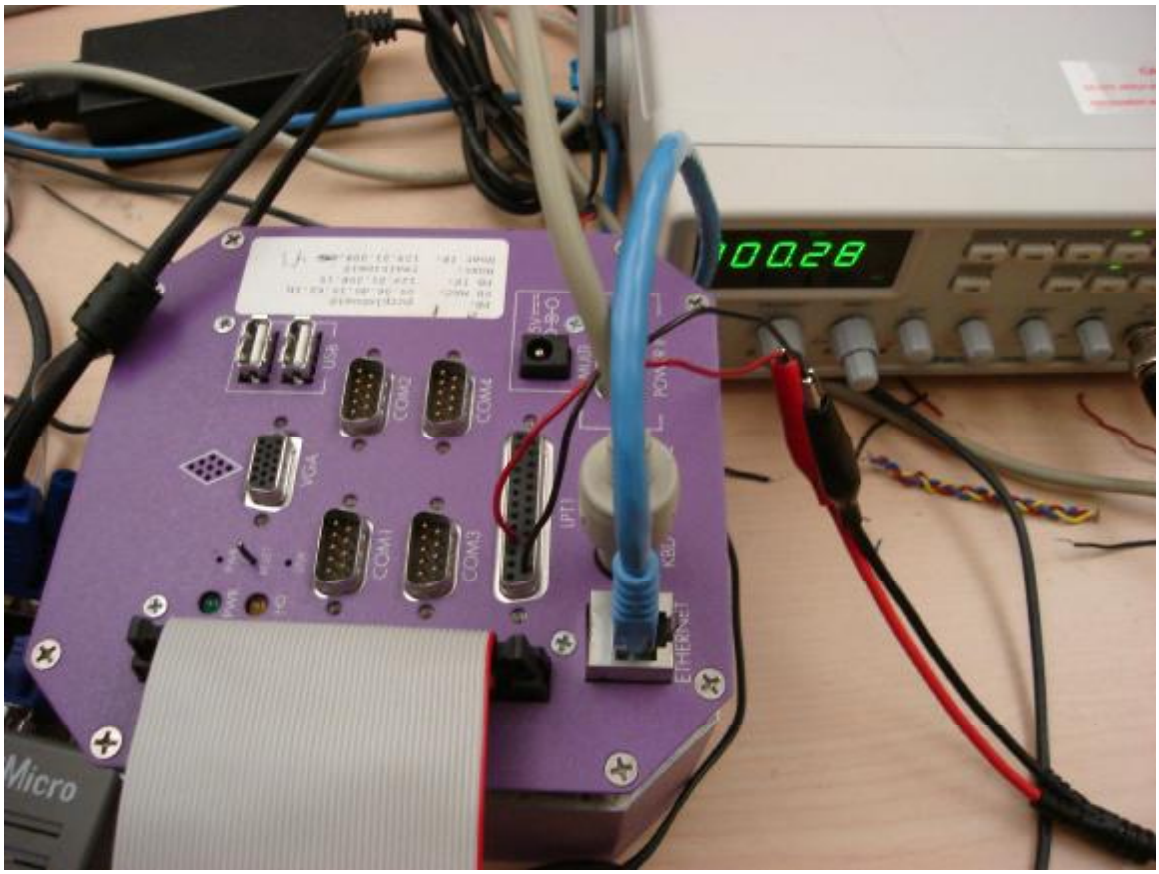
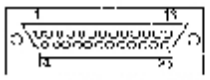
Interrupt Tutorial



The interrupt allows the programmer to immediately stop and perform time critical functions. The signal generator is used in this case to create interrupts at regular intervals, but any source can be used.



View is looking at
Connector side of
DB-25 Male Connector.

Pin	Description	
1	Strobe	PC Output
2	Data 0	PC Output
3	Data 1	PC Output
4	Data 2	PC Output
5	Data 3	PC Output
6	Data 4	PC Output
7	Data 5	PC Output
8	Data 6	PC Output
9	Data 7	PC Output
10	ACK	PC Input
11	Busy	PC Input
12	Paper Empty	PC Input
13	Select	PC Input
14	Auto Feed	PC Output
15	Error	PC Input
16	Initialize Printer	PC Output
17	Select Input	PC Output

Pin Assignments

Note: 8 Data Outputs
4 More Other Outputs

5 Data Inputs

Note: Pins 18-25 are
Ground

The above diagram is taken from the hardware manual and shows the pin out for the LPT1 printer port. Pin 10 will be used for the interrupts. The source of the interrupt needs to be connected to pin 10 and the ground needs to be connected to any pin between 18 and 25. The next step is to setup the software on the purple box to read the interrupt.

```
#define PORT_LENGTH           1     /* single register */
#define DATA_ADDRESS          0x378
#define STATUS_ADDRESS        0x379
#define CTRL_ADDRESS          0x37a
#define INIT_BIT              0x04
#define INTR_BIT              0x10
#define PARALLEL_IRQ          0x07 /* parallel port's interrupt vector
*/

/* bit 2 = printer initialization (high to initialize) */
/* bit 4 = hardware IRQ (high to enable) */
#define INIT_BIT              0x04
#define INTR_BIT              0x10

volatile unsigned _pulseCount;
uintptr_t ctrl_handle;
int interruptID;
```

The following segment of code will give later code, the ability to access the hardware.

```
int privity_err;
privity_err = ThreadCtl( _NTO_TCTL_IO, NULL );
if ( privity_err == -1)
{
      printf( "Can't get root permissions\n");
      return -1;
}
```

The following segment will get a pointer to the memory address of the control address for the printer port.

```
// Get a handle to the parallel port's Control Register
ctrl_handle = mmap_device_io( PORT_LENGTH, CTRL_ADDRESS );
if ( ctrl_handle == MAP_DEVICE_FAILED ) {
      perror( "control map failed" );
      exit(EXIT_FAILURE);
}
```

Standard Parallel Port Control Register		
Base	378	
Location	Base+2	37A
Bits	Bit 0	Strobe
	Bit 1	Auto Linefeed
	Bit 2	Initialize Printer
	Bit 3	Select Printer
	Bit 4	Enable IRQ Via Ack Line
	Bit 5	Enable bi-directional port
	Bit 6	Unused
	Bit 7	Unused

The following segment initializes and enables the interrupts on that port. The out8() function writes a one to the second bit in the register which initializes the printer. Then the next out8() function call writes a one to the fourth bit which enables interrupts on the ACK line.

```
out8( ctrl_handle, INIT_BIT );
out8( ctrl_handle, INTR_BIT );
```

The following method, InterruptAttach(), is used to used to setup the interrupt handler. The first argument that is passed in defines the Interrupt that is to be attached, in this case it our interrupt is defines as PARALLEL_IRQ. The next argument is the function that handles the interrupt which is interruptReceived in our case. The function needs to be setup with two input arguments, here they are the *arg and the id. The next argument passed into InterruptAttach is size which indicates how big the data, in bytes, is that will be passed in arg of your function that handles the interrupt. The last argument to be passed is the flags parameter which is zero in our case. The interrupted is then checked to make sure that the function was performed correctly.

```
interruptID = InterruptAttach(PARALLEL_IRQ, interruptReceived, this,
sizeof(this), 0);
if (interruptID == -1) {
      fprintf(stderr, "can't attach to IRQ %d\n", PARALLEL_IRQ);
      perror(NULL);
      exit(EXIT_FAILURE);
}

const struct sigevent *
interruptReceived(void *arg, int id)
{
```

```
        atomic_add_value( &_pulseCount, 1 );
        return NULL;
}
```

With the code from above we have a program which creates interrupts when a signal is received from the ACK pin of the printer port and counts the number of pulses from the signal generator.